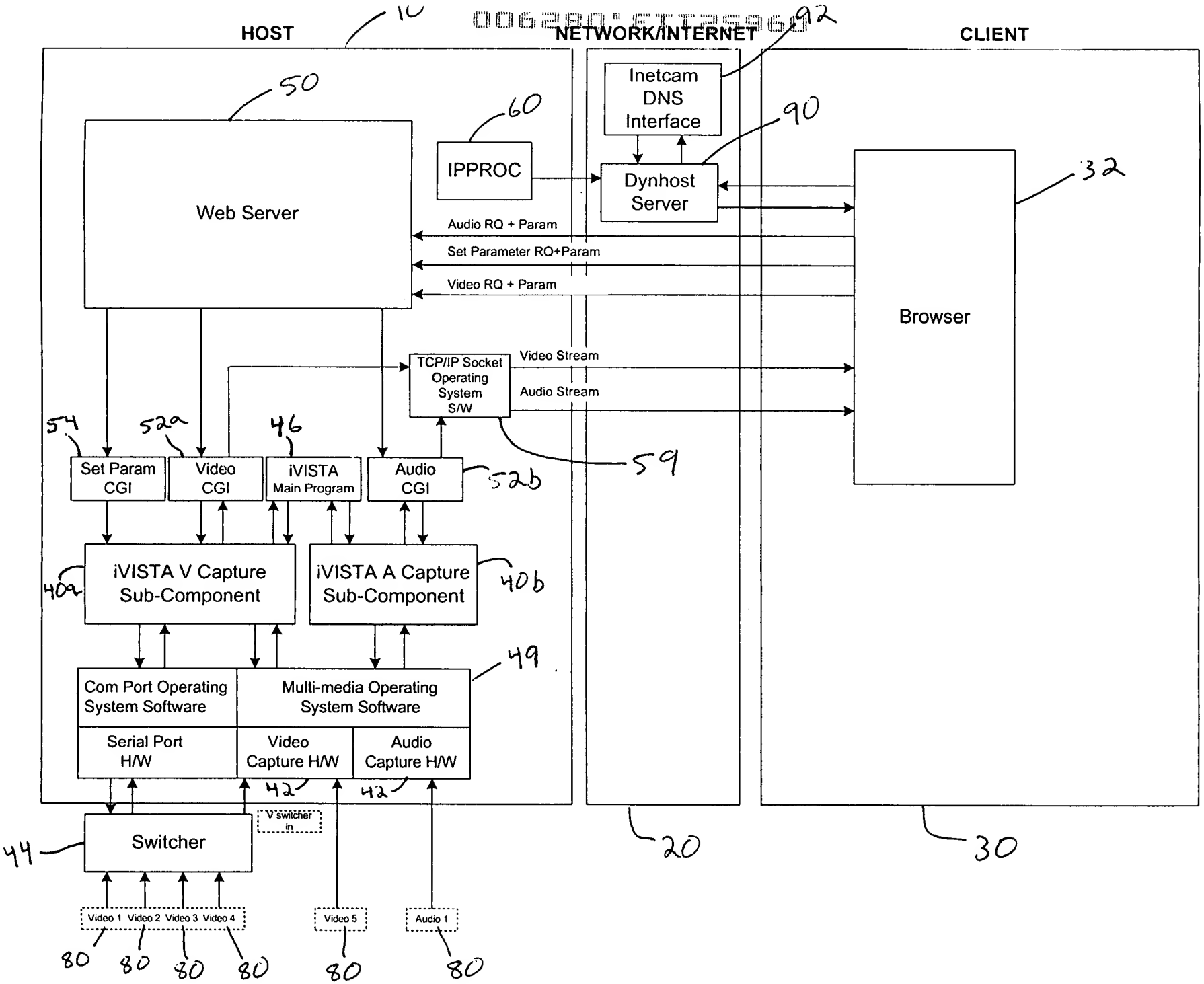

HOST
10
NETWORK/INTERNET
92
CLIENT
50
60
Inetcam DNS Interface
90
IPPROC
Web Server
Dynhost Server
32
Audio RQ + Param
Set Parameter RQ+Param
Video RQ + Param
Browser
TCP/IP Socket Operating System S/W
Video Stream
Audio Stream
59
54
52a
46
Set Param CGI
Video CGI
iVISTA Main Program
Audio CGI
52b
iVISTA V Capture Sub-Component
40a
iVISTA A Capture Sub-Component
40b
49
Com Port Operating System Software
Multi-media Operating System Software
Serial Port H/W
Video Capture H/W
Audio Capture H/W
42
42
V switcher in
Switcher
44
20
30
Video 1
Video 2
Video 3
Video 4
Video 5
Audio 1
80
80
80
80
80
80

Figure 1

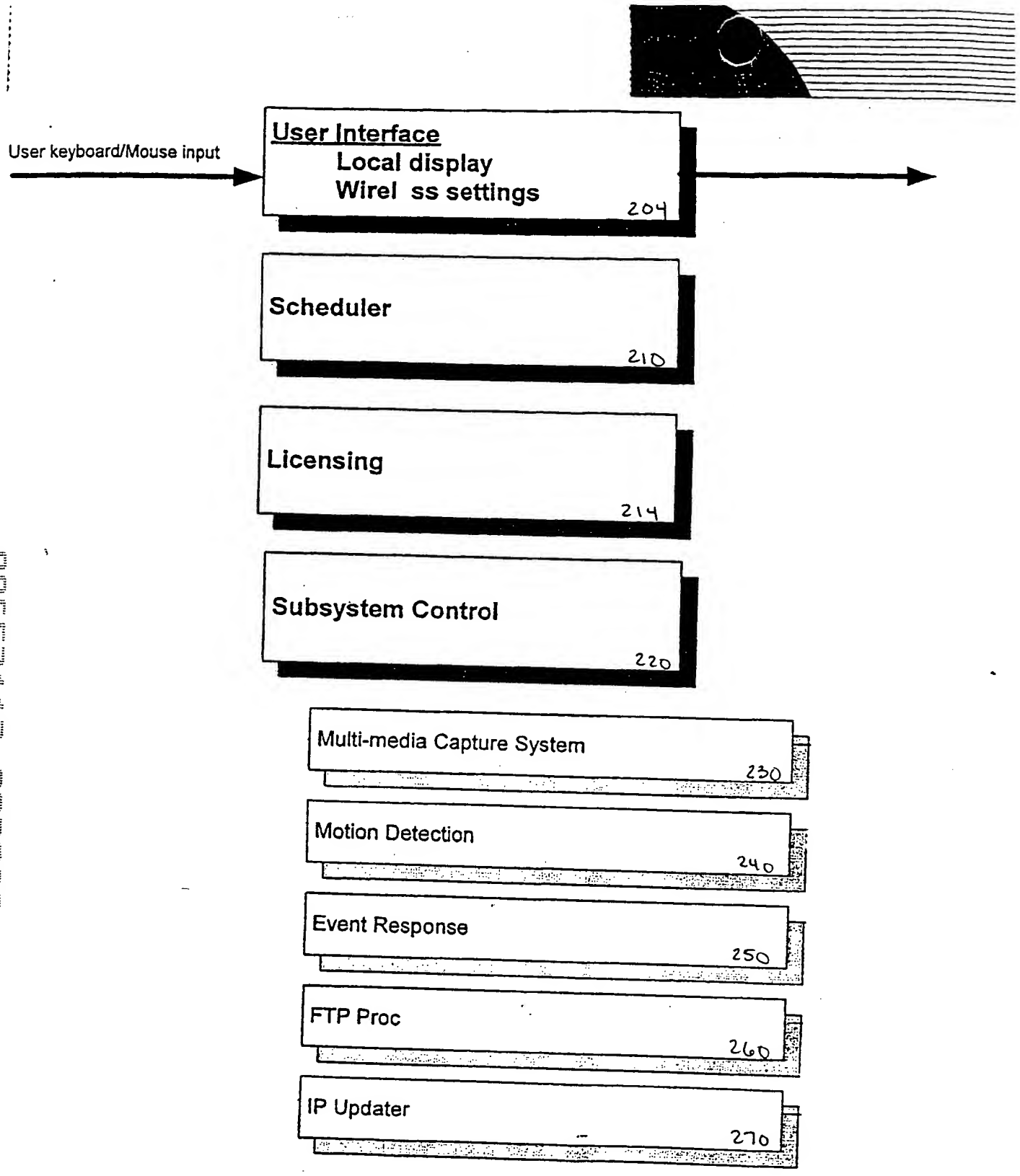
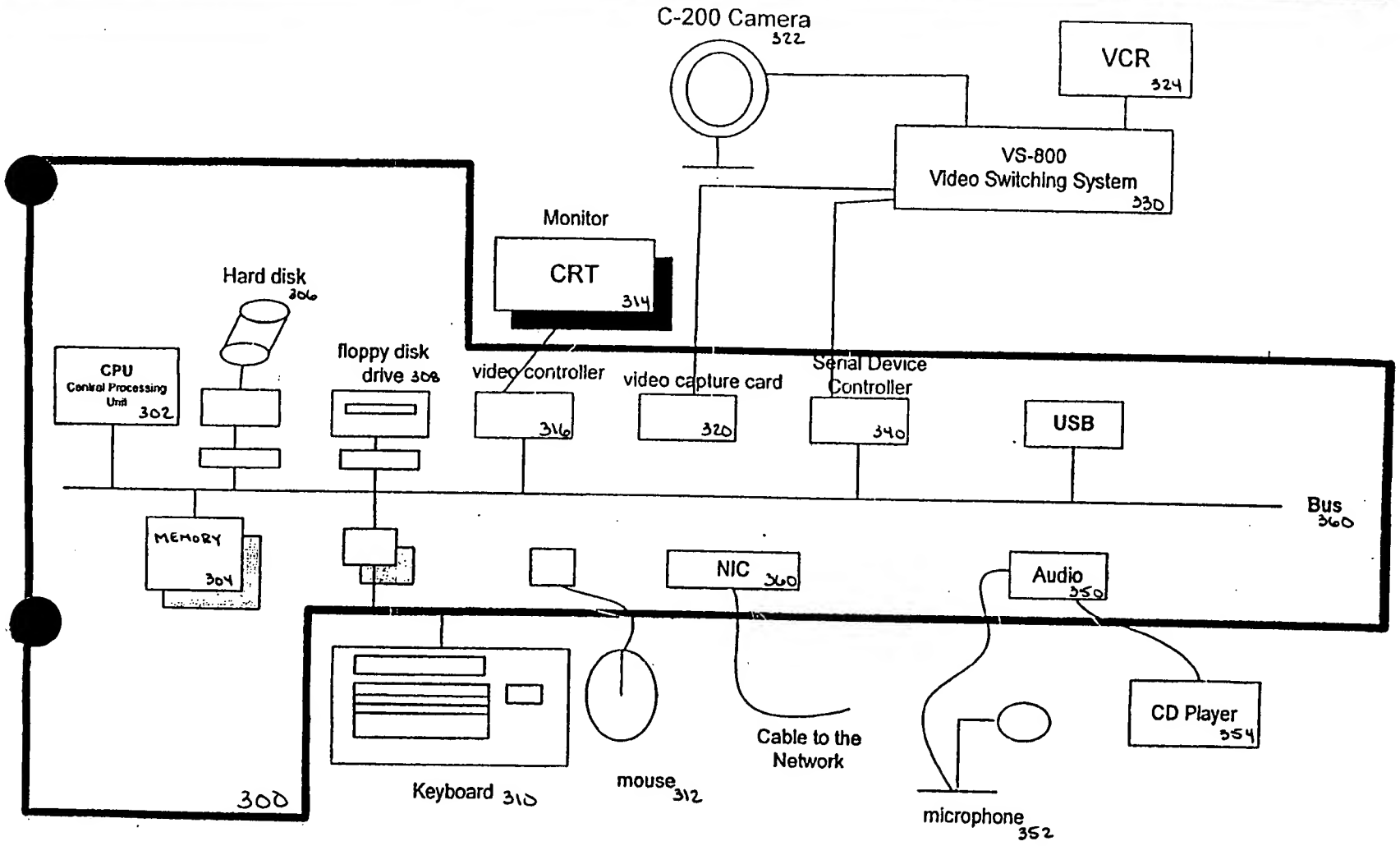

User keyboard/Mouse input
User Interface
Local display
Wirel ss settings
204
Scheduler
210
Licensing
214
Subsystem Control
220
Multi-media Capture System
230
Motion Detection
240
Event Response
250
FTP Proc
260
IP Updater
270

Fig 2



NIC = Network Interface Card

Fig 3

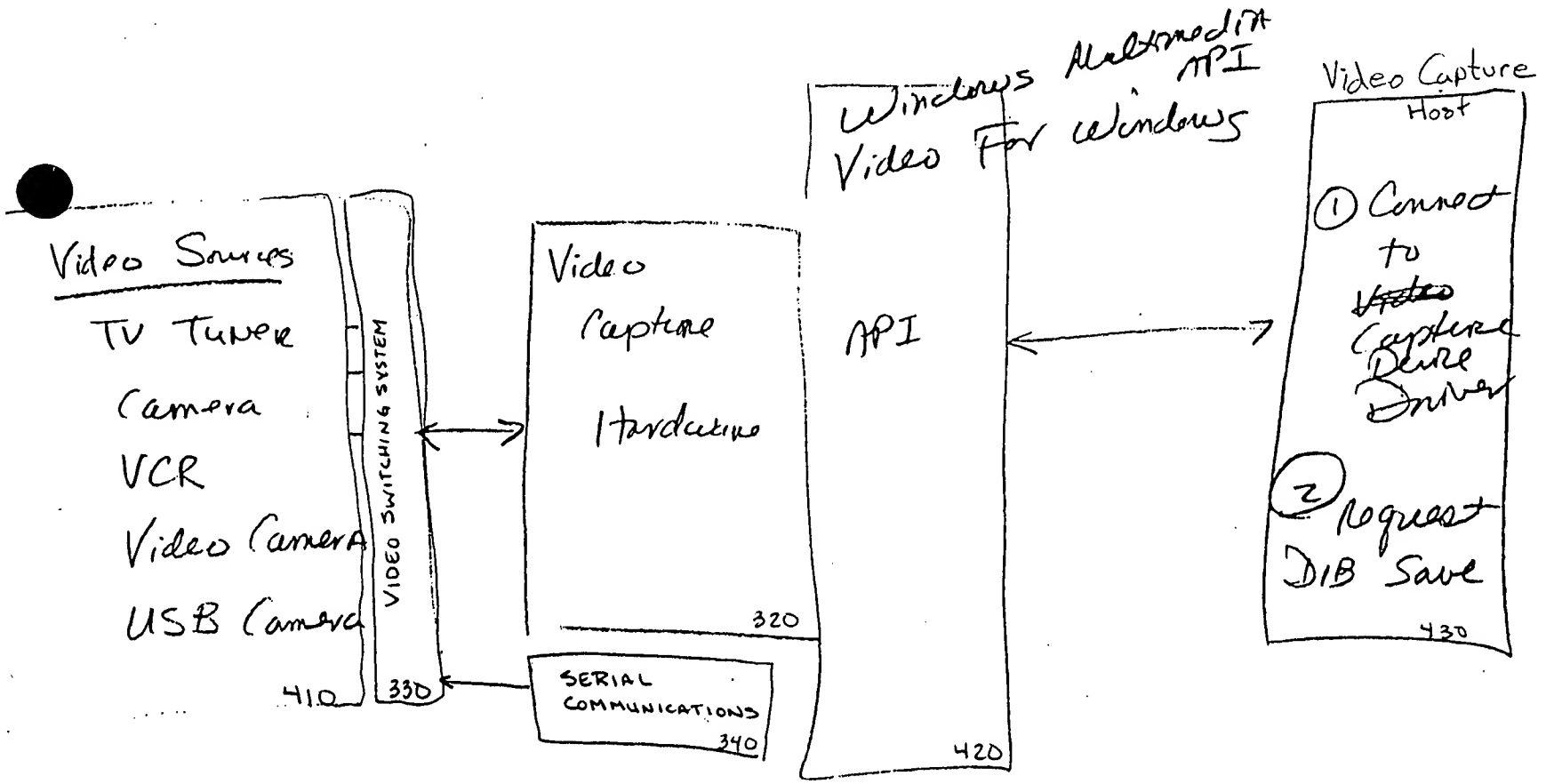

Video Sources
TV Tuner
Camera
VCR
Video Camera
USB Camera
410
VIDEO SWITCHING SYSTEM
330
Video
Capture
Hardware
320
SERIAL
COMMUNICATIONS
340
Windows Multimedia
API
Video For Windows
API
420
Video Capture
Host
1 Connect
to
Capture
Device
Driver
2 Request
DIB Save
430

Fig 4A

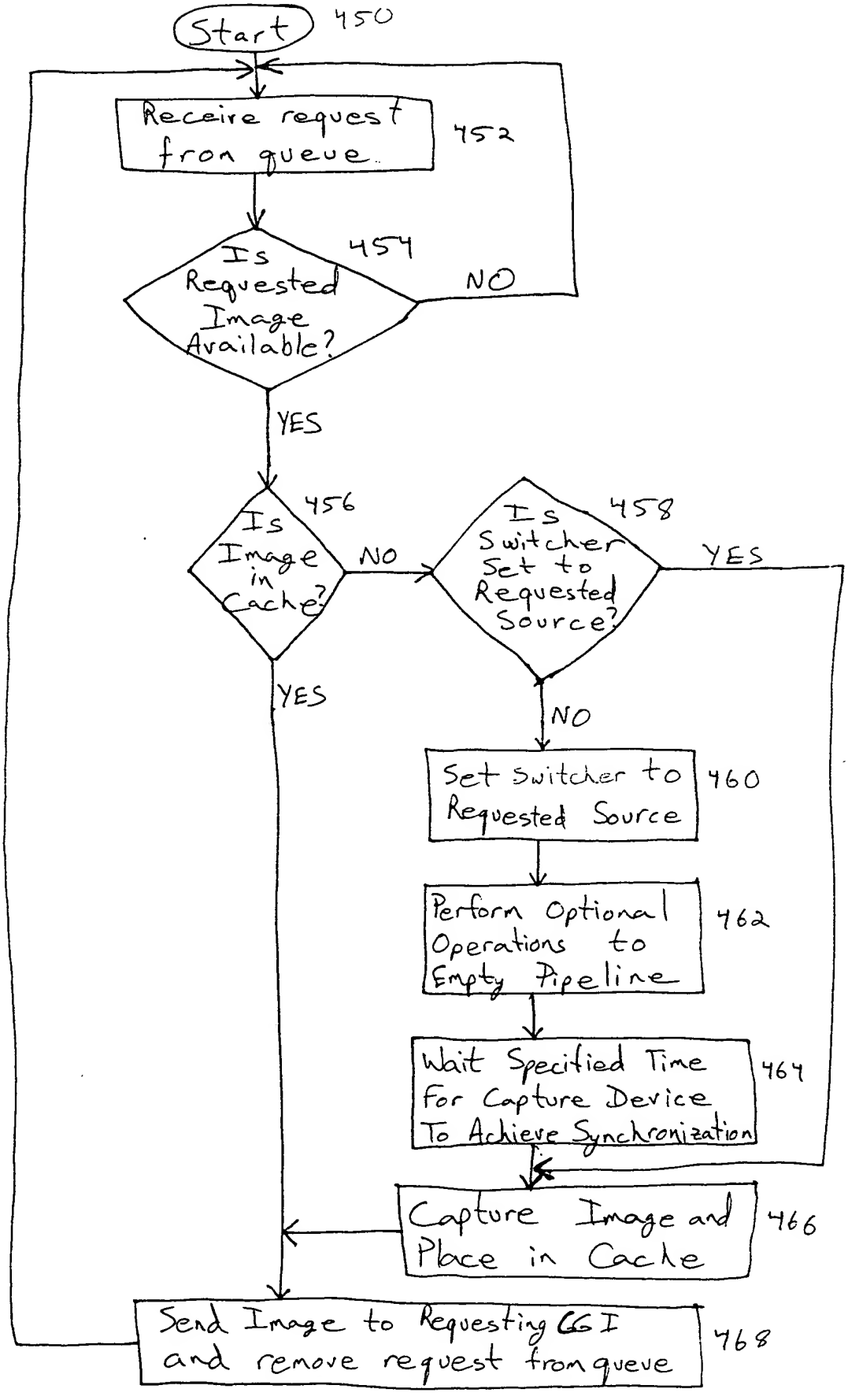

Start
450
Receive request from queue
452
Is Requested Image Available?
454
NO
YES
Is Image in Cache?
456
NO
Is Switcher Set to Requested Source?
458
YES
YES
NO
Set switcher to Requested Source
460
Perform Optional Operations to Empty Pipeline
462
Wait Specified Time For Capture Device To Achieve Synchronization
464
Capture Image and Place in Cache
466
Send Image to Requesting CGI and remove request from queue
468

Figure 4B

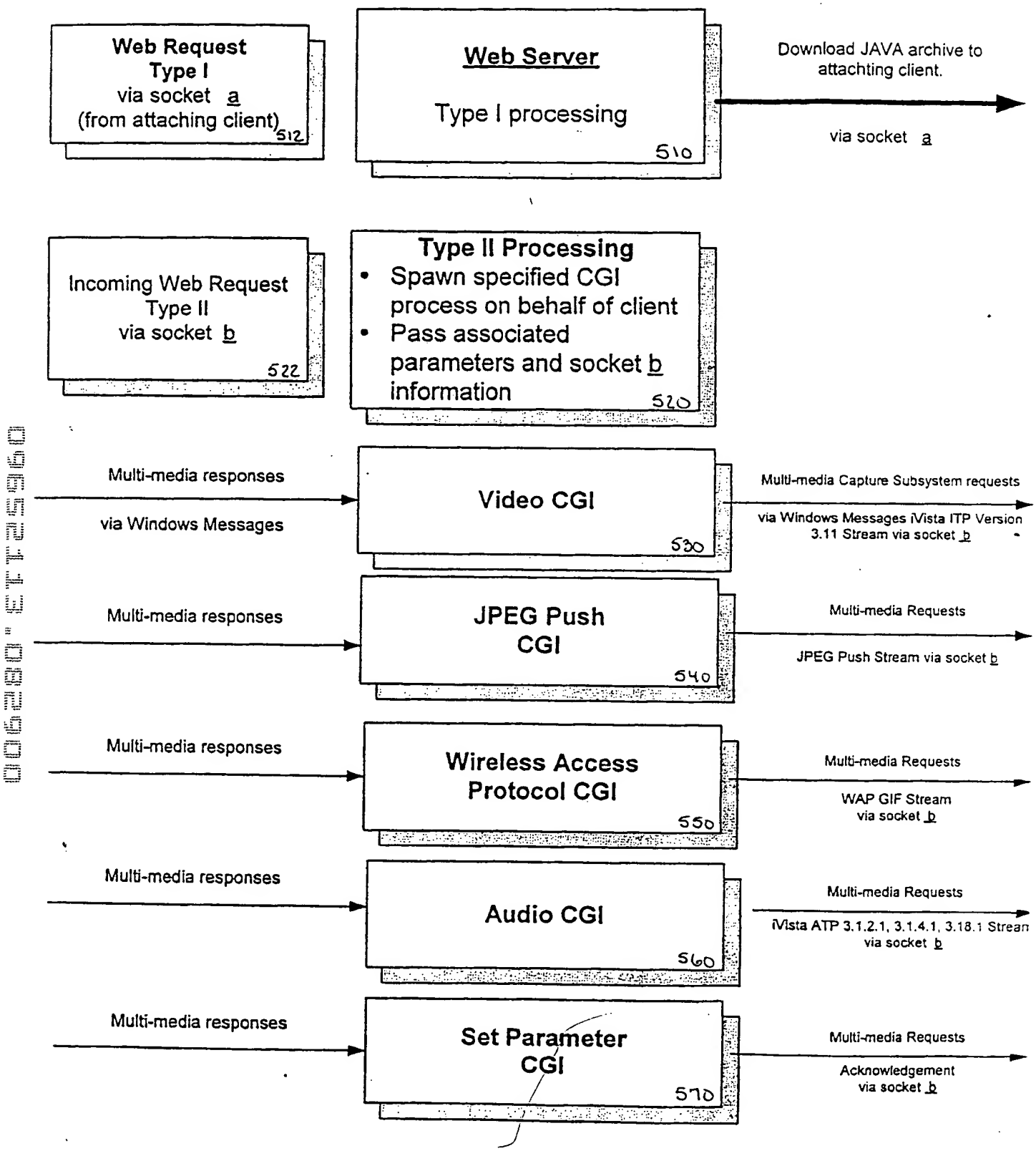

Web Request
Type I
via socket a
(from attaching client)
512
Web Server
Type I processing
510
Download JAVA archive to attachting client.
via socket a
Incoming Web Request
Type II
via socket b
522
Type II Processing
• Spawn specified CGI process on behalf of client
• Pass associated parameters and socket b information
520
Multi-media responses
via Windows Messages
Video CGI
530
Multi-media Capture Subsystem requests
via Windows Messages iVista ITP Version 3.11 Stream via socket b
Multi-media responses
JPEG Push CGI
540
Multi-media Requests
JPEG Push Stream via socket b
Multi-media responses
Wireless Access Protocol CGI
550
Multi-media Requests
WAP GIF Stream via socket b
Multi-media responses
Audio CGI
560
Multi-media Requests
iVista ATP 3.1.2.1, 3.1.4.1, 3.18.1 Stream via socket b
Multi-media responses
Set Parameter CGI
570
Multi-media Requests
Acknowledgement via socket b

Fig. 5A

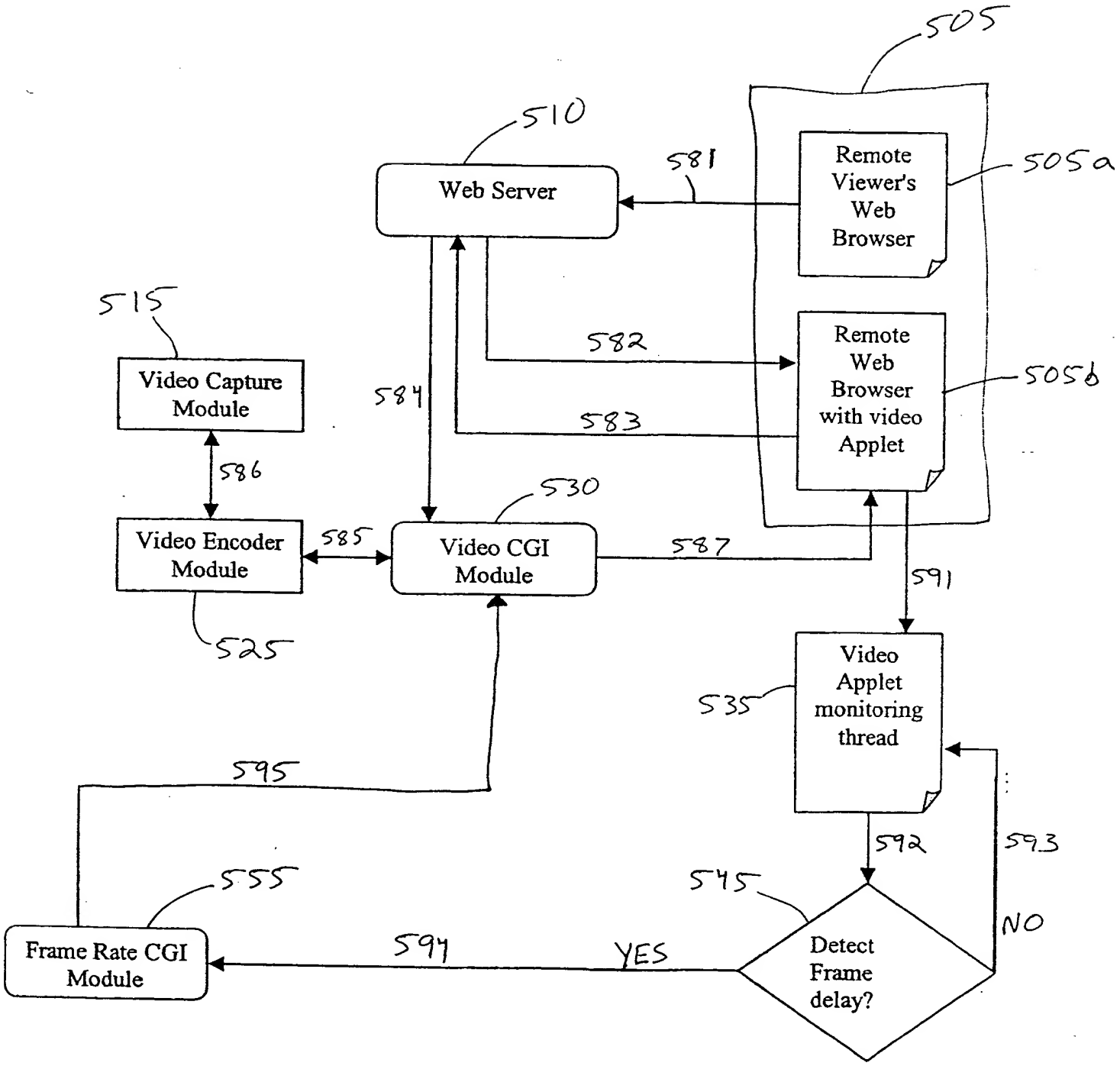
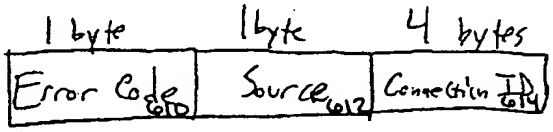

505
510
Web Server
581
Remote Viewer's Web Browser
505a
515
Video Capture Module
582
584
583
Remote Web Browser with video Applet
505b
586
530
Video Encoder Module
585
Video CGI Module
587
591
525
Video Applet monitoring thread
535
595
592
593
555
545
Frame Rate CGI Module
594
YES
Detect Frame delay?
NO

Fig. 5B

Video Stream Format

Video Header

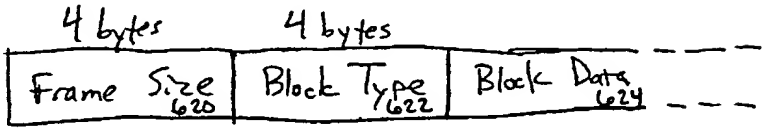


0 = success
non-zero = error

0 = host
1 = mirror

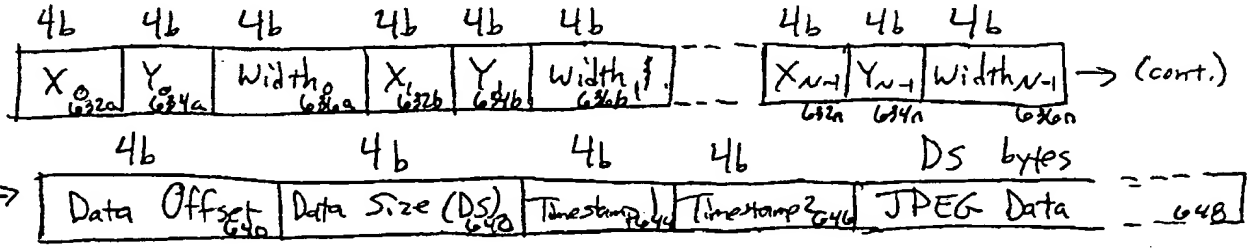
Note: If Error Code is non-zero, no bytes will follow and the stream is complete.

Video Block



The following are the possible block types and their block data formats

Block Type = N (N > 0), Partial Frame



In this case, the image consists of N segments arranged in a horizontal "stripe". The $(X_k, Y_k, Width_k)$ triplets describe the destination position and width of each segment. Each segment is 16 pixels tall.

Figure: (A "stripe" image)

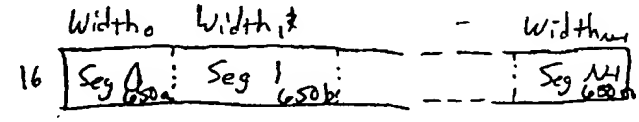
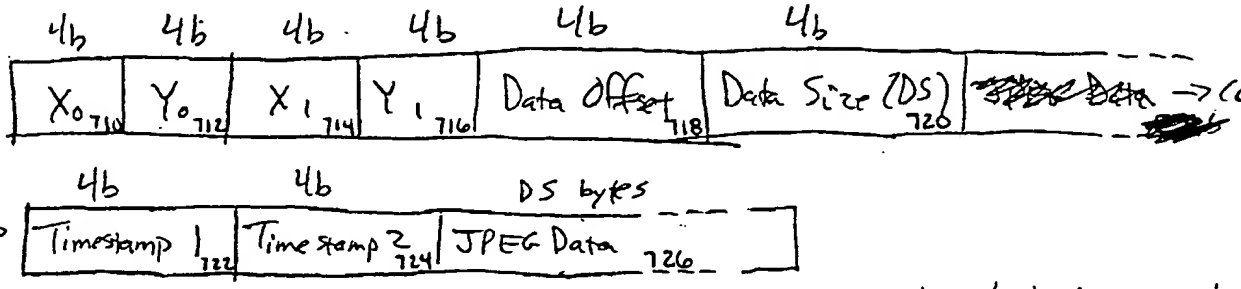


Fig 6

Block Type = -3 , Single Block

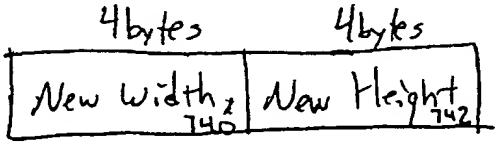


In this case, the JPEG is a single rectangle which is moved to (X_0, Y_0)–(X_1, Y_1) in the destination image.

Block Type = -4, Synchronization Frame

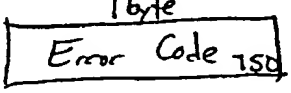
The format of this Block is identical to the Single Block described above with $X_0 = 0$, $Y_0 = 0$, X_1 = Width of Image, Y_1 = Height of Image. The block is used to resynchronize the video stream with real time.

Block Type = -1, New Image Size



This block indicates a change in the transmitted image size. It is immediately followed by a full image Single Block frame.

Block Type = -2, Error Block



This block indicates an error in the stream. The transmission is terminated following the error code.

Fig. 7

Figure 8

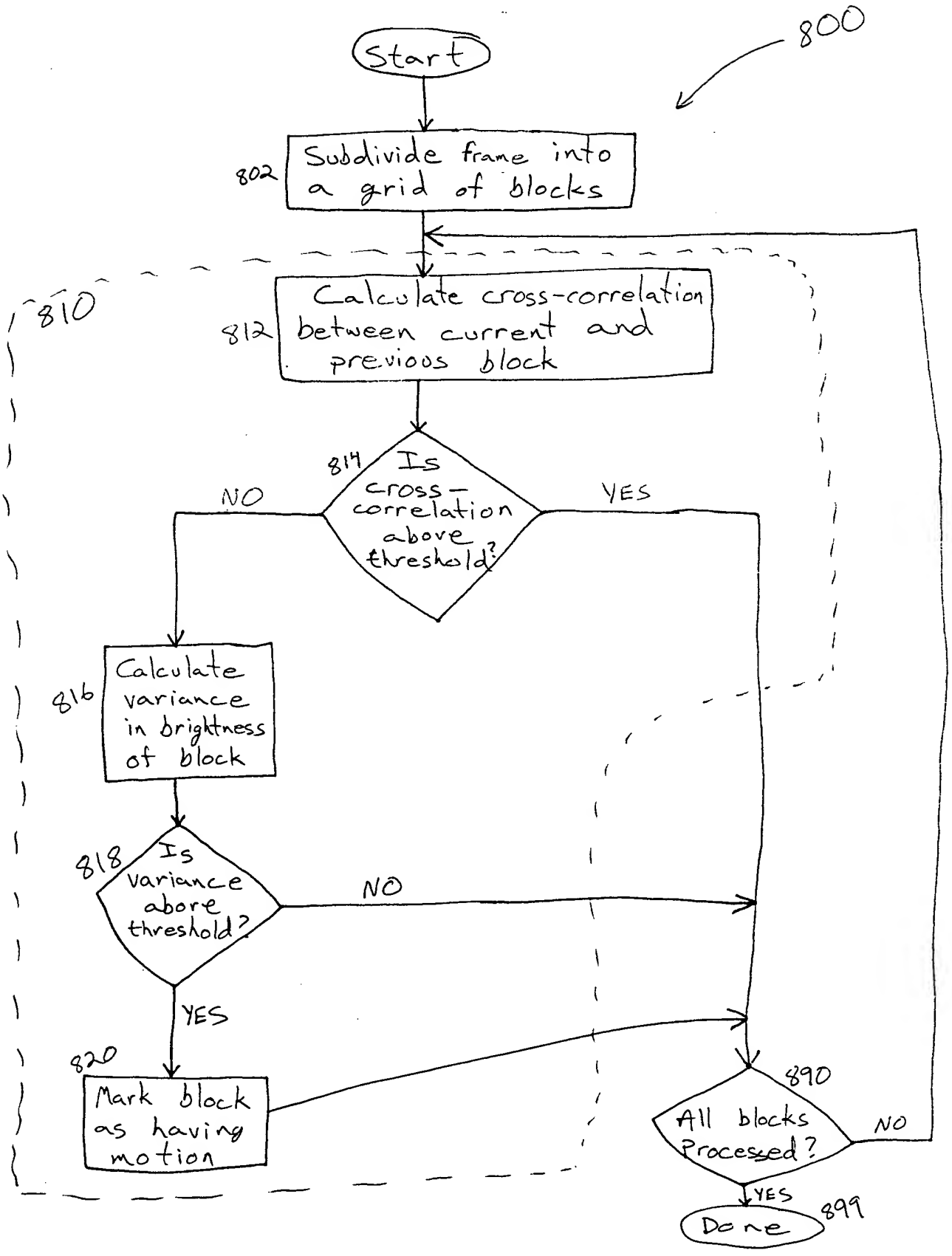

800
Start
802
Subdivide frame into a grid of blocks
810
812
Calculate cross-correlation between current and previous block
814
Is cross-correlation above threshold?
NO
YES
816
Calculate variance in brightness of block
818
Is variance above threshold?
NO
YES
820
Mark block as having motion
890
All blocks Processed?
NO
YES
899
Done

Figure 9

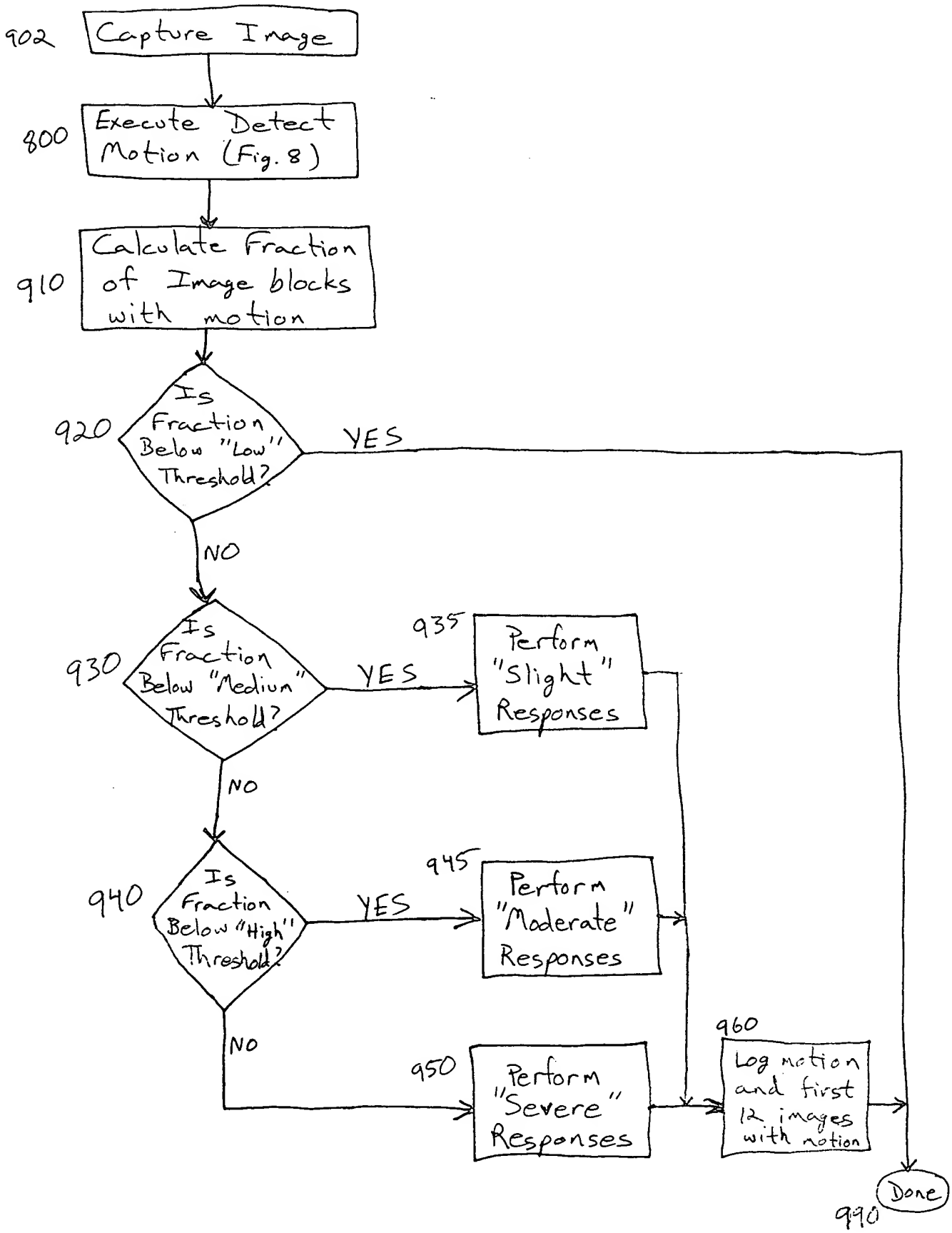
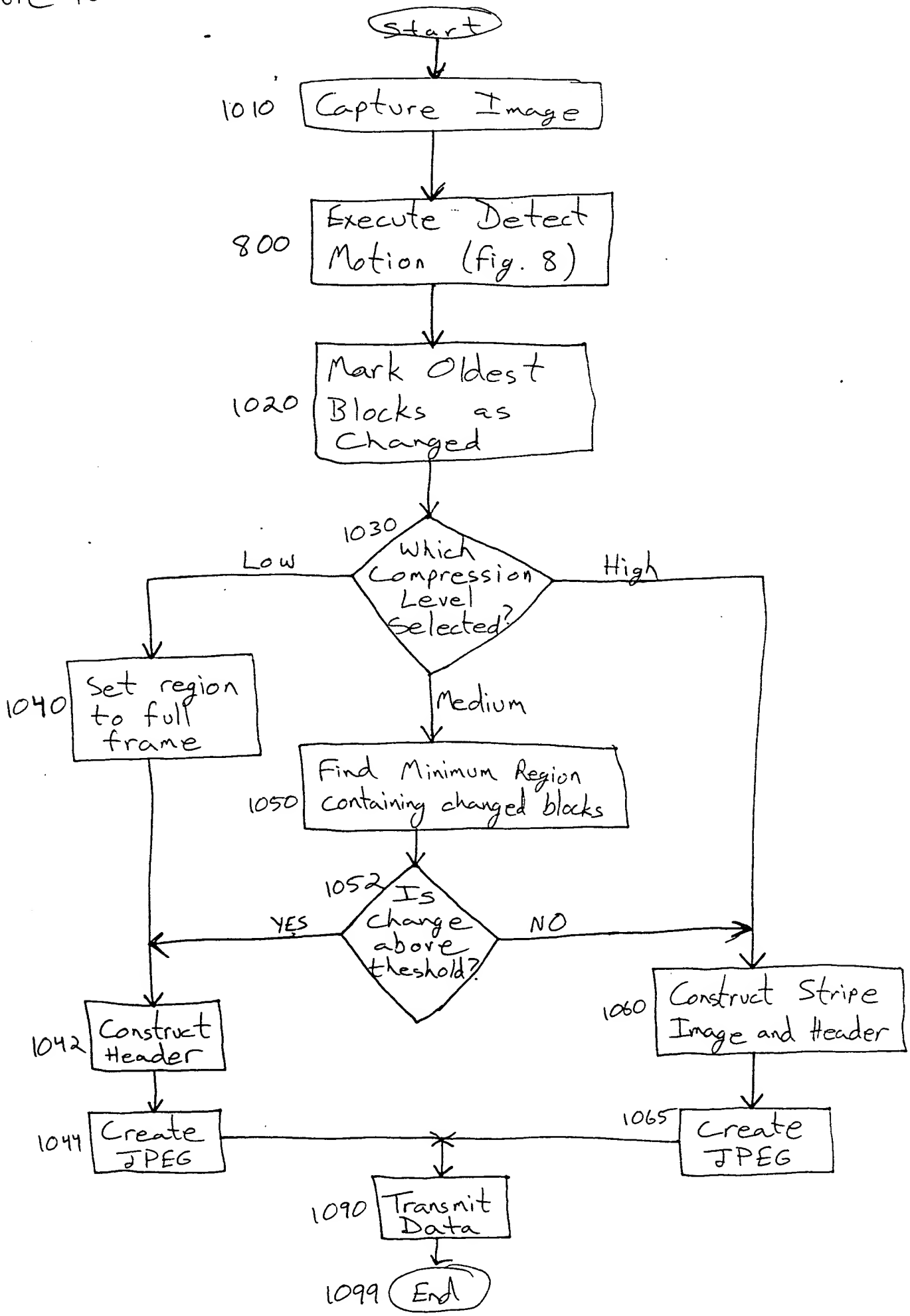
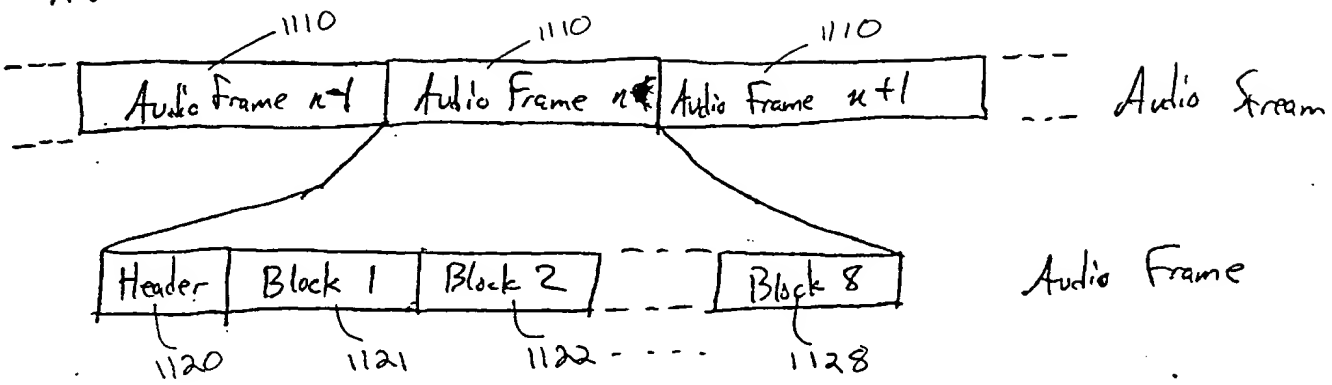

902
Capture Image
800
Execute Detect Motion (Fig. 8)
910
Calculate Fraction of Image blocks with motion
920
Is Fraction Below "Low" Threshold?
YES
NO
930
Is Fraction Below "Medium" Threshold?
YES
935
Perform "Slight" Responses
NO
940
Is Fraction Below "High" Threshold?
YES
945
Perform "Moderate" Responses
NO
950
Perform "Severe" Responses
960
Log motion and first 12 images with motion
990
Done

Figure 10

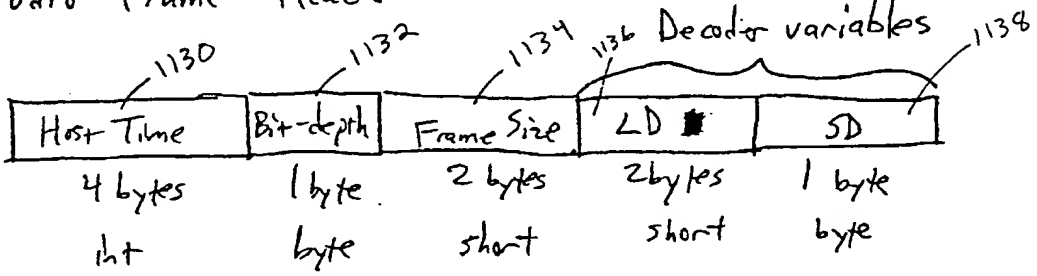

Start
1010
Capture Image
800
Execute Detect Motion (fig. 8)
1020
Mark Oldest Blocks as Changed
1030
Which Compression Level Selected?
Low
High
Medium
1040
Set region to full frame
1050
Find Minimum Region containing changed blocks
1052
Is change above theshold?
YES
NO
1042
Construct Header
1044
Create JPEG
1060
Construct Stripe Image and Header
1065
Create JPEG
1090
Transmit Data
1099
End

Stream

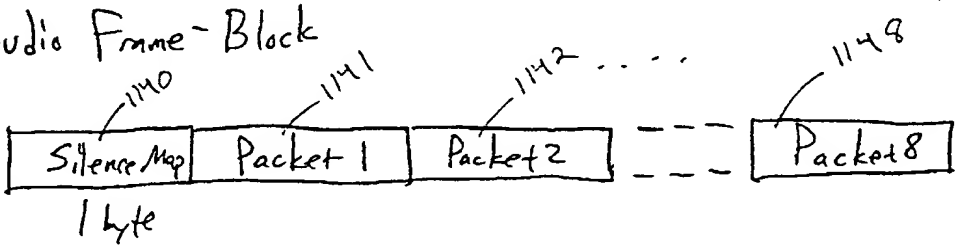
Audio Format



Audio Frame - Header



Audio Frame - Block



Each bit in the silence map indicates whether the corresponding packet exists in the stream.

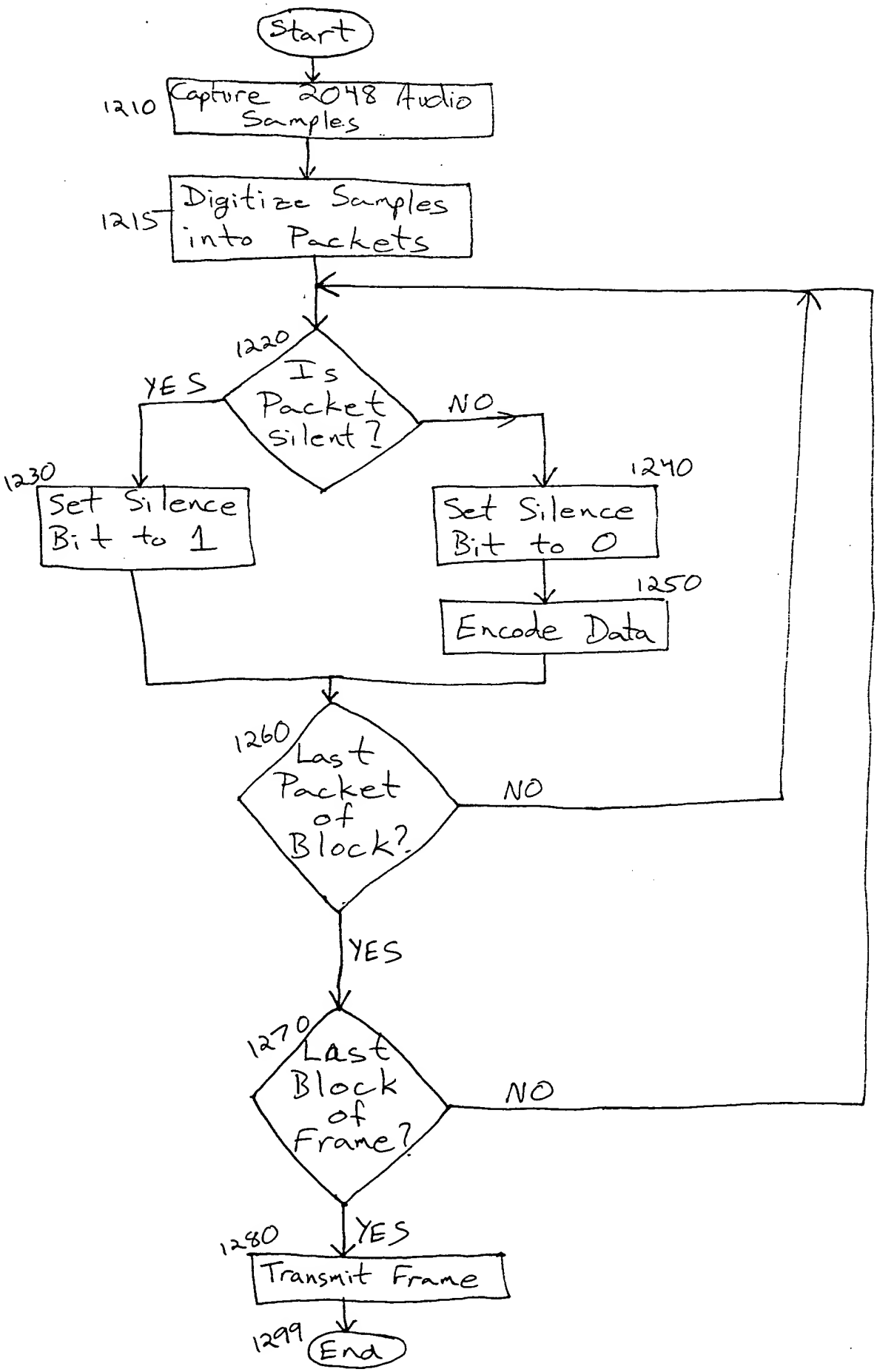
Each packet consists of 32 samples of 2-, 4-, or 8-bits each (thus 8, 16, or 32 bytes per packet).

2- and 4-bit encodings are in ADPCM format.

8-bit encoding is μ-law compressed samples.

Fig. 11

Figure 12


Start
1210
Capture 2048 Audio Samples
1215
Digitize Samples into Packets
1220
Is Packet Silent?
YES
NO
1230
Set Silence Bit to 1
1240
Set Silence Bit to 0
1250
Encode Data
1260
Last Packet of Block?
NO
YES
1270
Last Block of Frame?
NO
YES
1280
Transmit Frame
1299
End

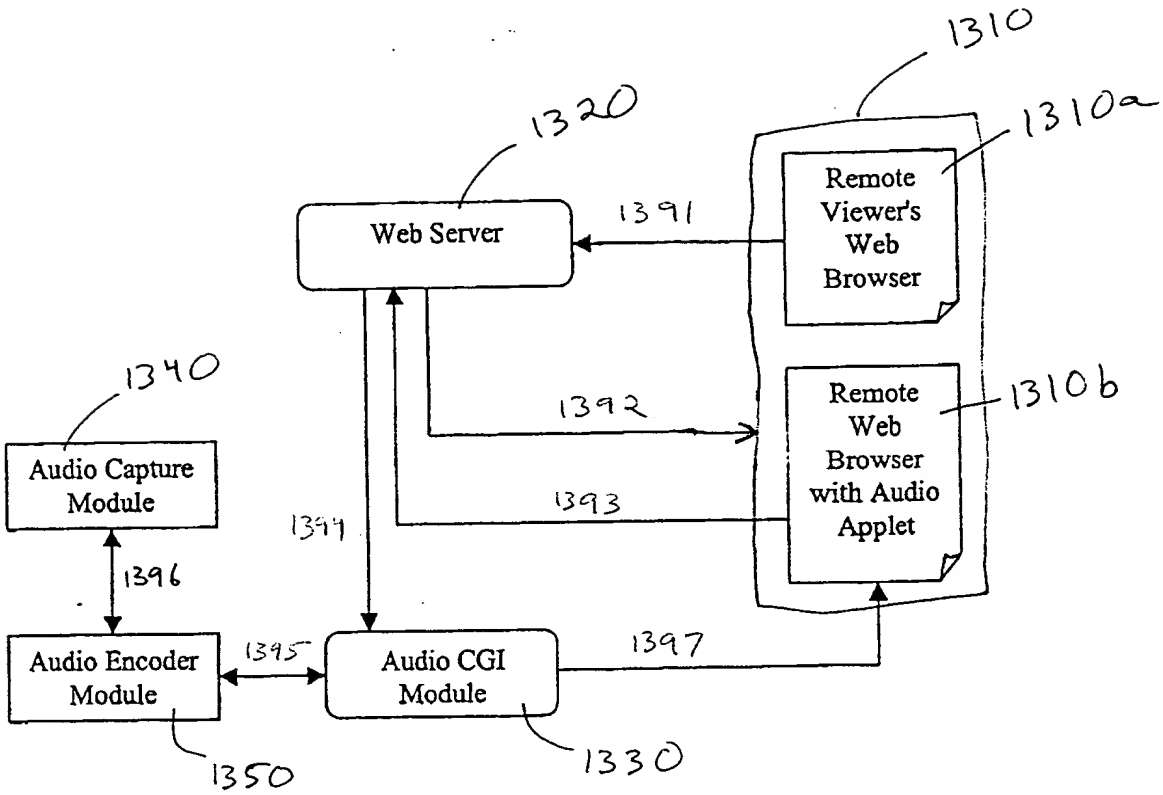

1310
1320
1310a
Web Server
1391
Remote Viewer's Web Browser
1340
1392
Remote Web Browser with Audio Applet
1310b
Audio Capture Module
1393
1394
1396
Audio Encoder Module
1395
Audio CGI Module
1397
1350
1330

Fig. 13

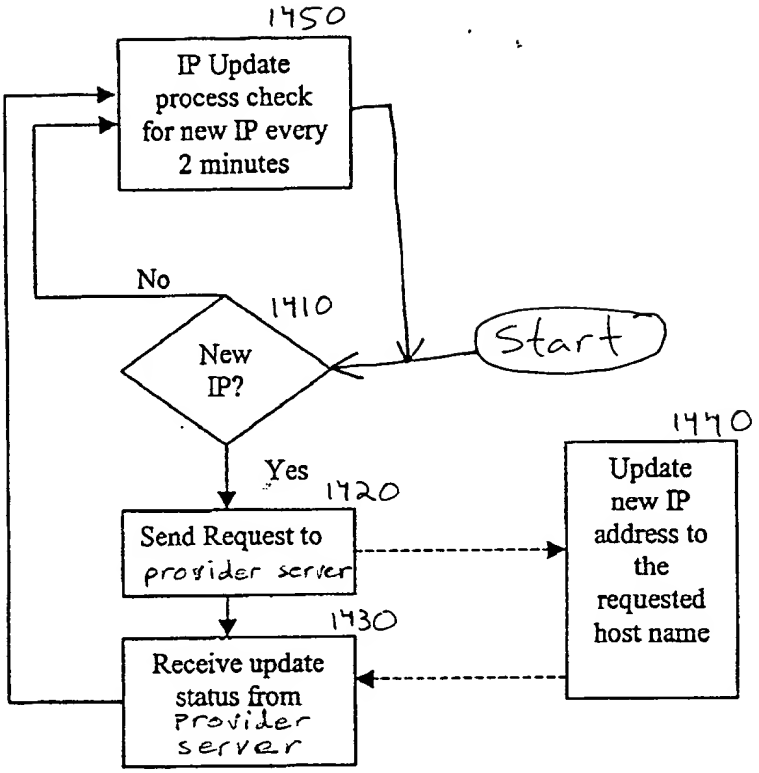

1450
IP Update process check for new IP every 2 minutes
No
1410
New IP?
Start
Yes
1420
Send Request to provider server
1440
Update new IP address to the requested host name
1430
Receive update status from provider server

Fig. 14

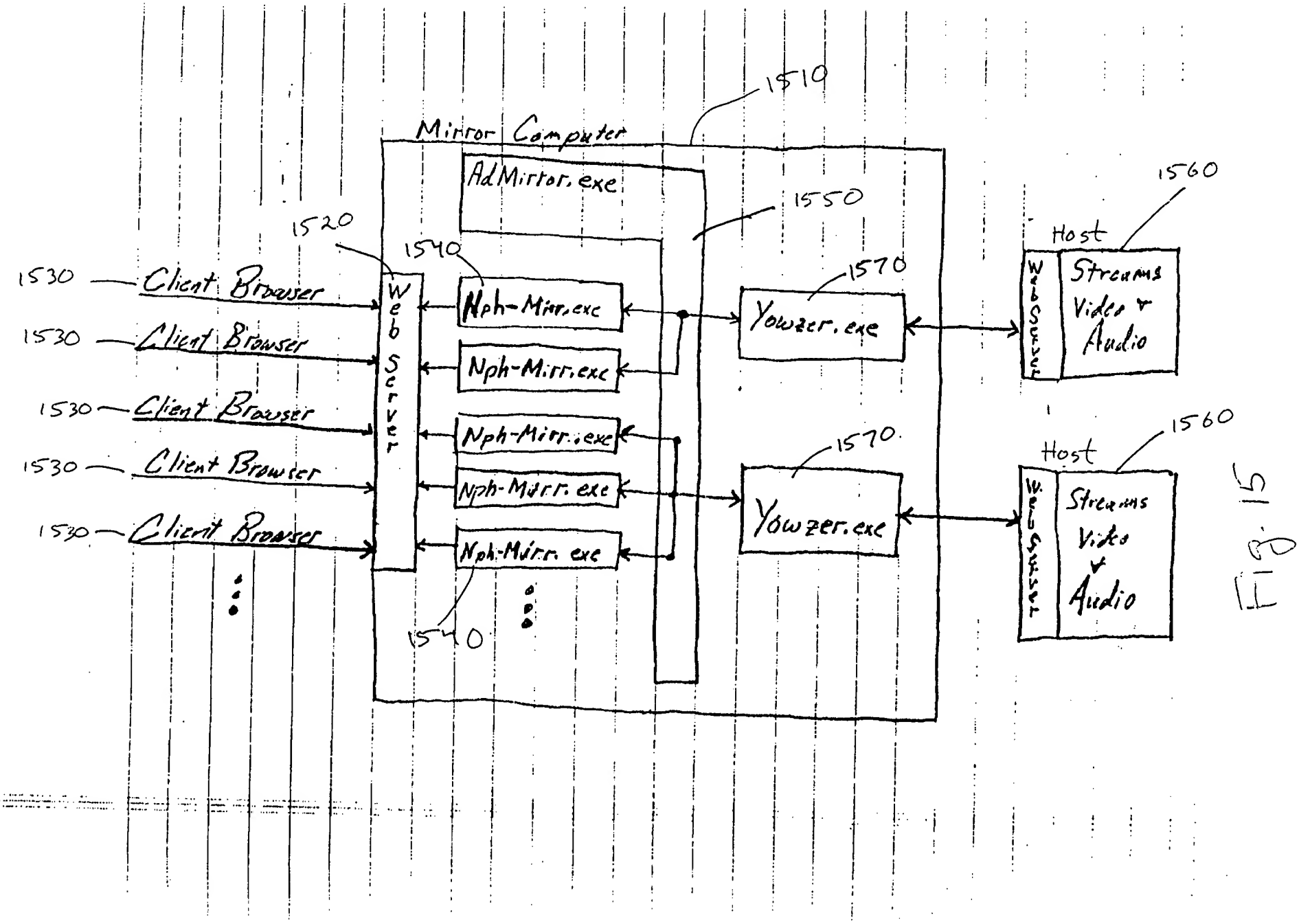

1510
Mirror Computer
AdMirror.exe
1550
1520
1540
1530
Client Browser
1530
Client Browser
1530
Client Browser
1530
Client Browser
1530
Client Browser
Web Server
Nph-Mirr.exe
Nph-Mirr.exe
Nph-Mirr.exe
Nph-Mirr.exe
Nph-Mirr.exe
1540
1570
Yowzer.exe
1570
Yowzer.exe
1560
Host
WebServer
Streams
Video
&
Audio
1560
Host
WebServer
Streams
Video
&
Audio

FIG. 15